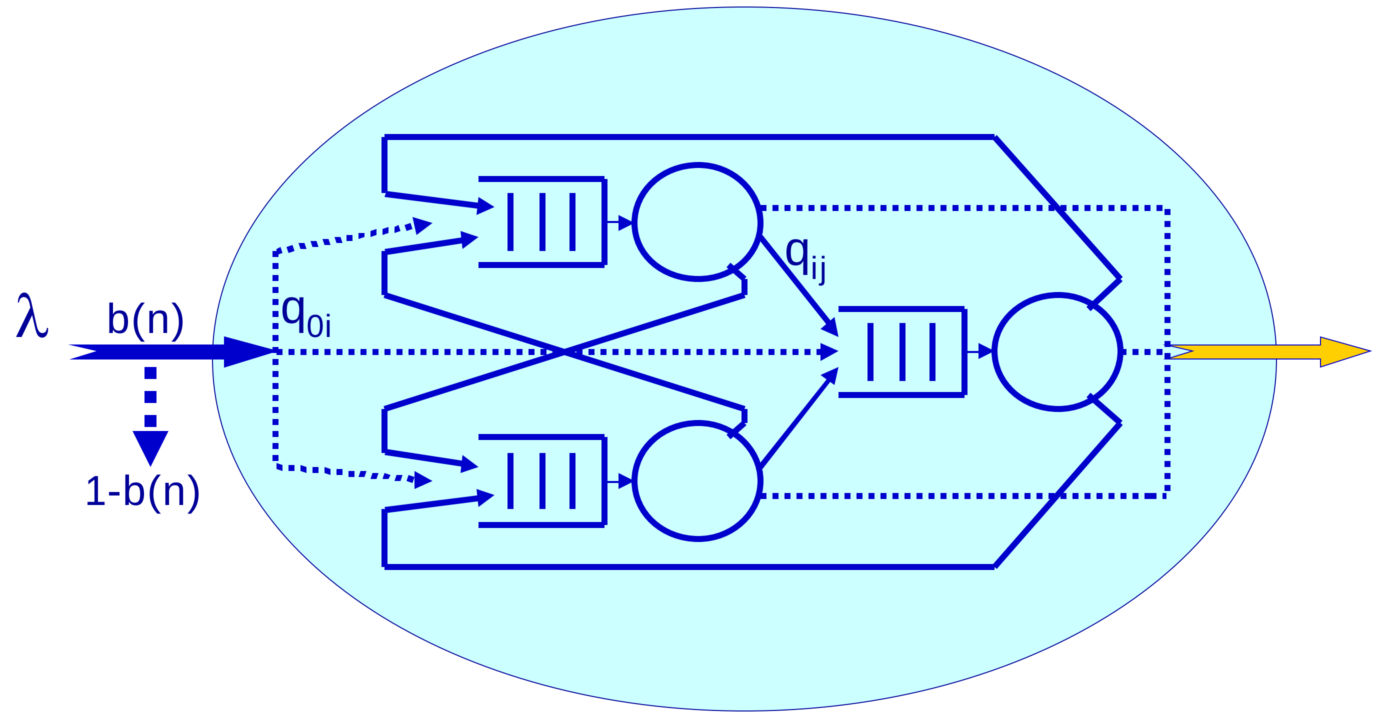
Event-Based Stochastic Learning and Optimization

Xi-Ren Cao

Hong Kong Uni. of Science & Tech.

Example 1: Admission Control in Communication



n: number of all customers in network

n_i: number of customers at server i

$n=(n_1,\ldots,n_M)$: state, N: Capacity

How do we choose the admission probability b(n)?

The Problem

- Non-standard
  - Not a Markov Decision Process (MDP)
    Action depends on the population n,
    Different state n need to take the same action
  - Not a Partially Observable MDP (POMDP)
    When a customer arrives, know something about
    the next state (population will not decrease)
- Additional New features
  - Control only applied when a customer arrives
  - Relatively small policy space: n -> *a* *vs* n -> *a*
- New formulation and approach
  - Event based and sensitivity view
    Not traditional: MDP with constraints
  - Generally applicable to many other problems

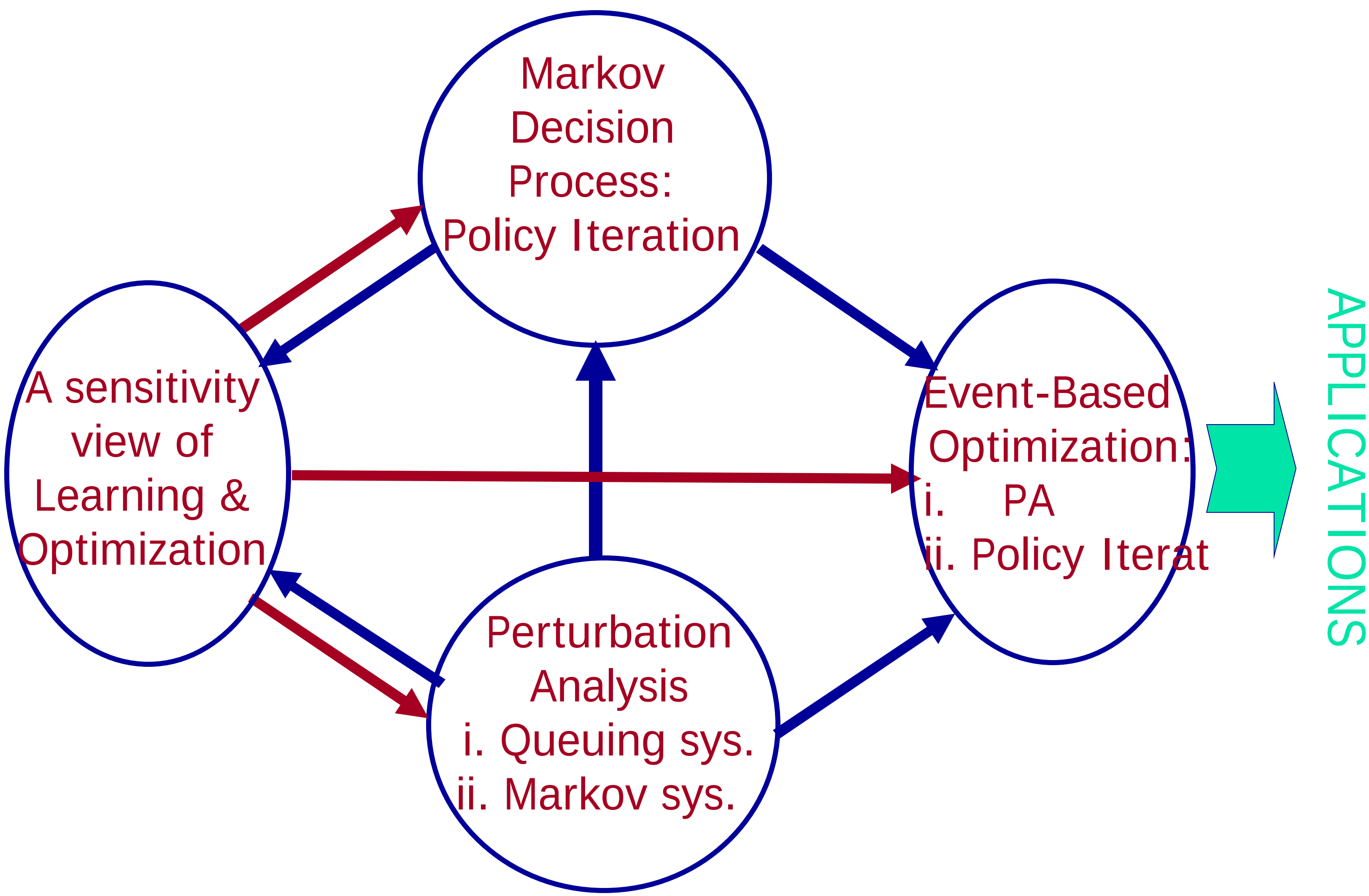
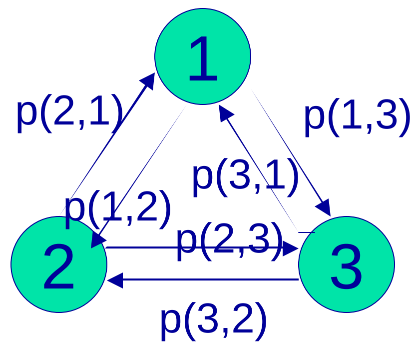

Markov Decision Process: Policy Iteration
A sensitivity view of Learning & Optimization
Event-Based Optimization:
i. PA
ii. Policy Iterat
Perturbation Analysis
i. Queuing sys.
ii. Markov sys.
APPLICATIONS

Table of Contents

Event-Based Approaches

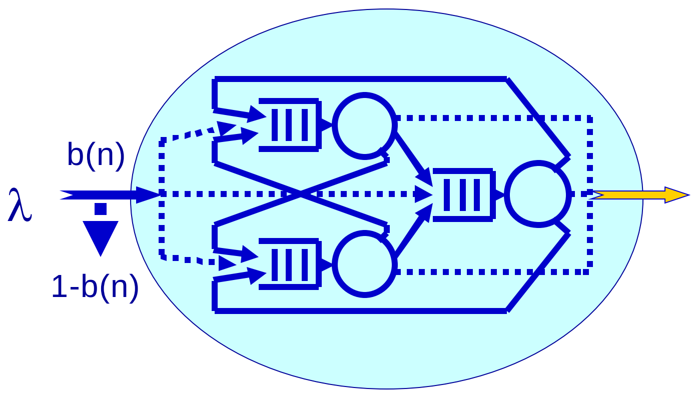
The Markov Model



- $X=\{X_n, n=1,2,.....\}$, ergodic, X_n in $S=\{1,2,...,M\}$.
- Transition prob. Matrix $P=[p(i,j)]_{ij=1,..,M}$
- Steady-state probability: $\pi=(\pi(1), \pi(2),..,\pi(M))$.
- Performance function: $f=(f(1), ..., f(M))^T$
- Steady-state performance: $\eta = \pi f = \Sigma_i \pi(i)f(i)$

$Pe=e$, $\pi(I-P)=0$, $\pi e=1$. $e=(1,1,\ldots,1)^T$, I: identity

Formulating Events



An event may contain information about current state, and next state after transition

A customer arrival: population not reduced after transition

A transition: $<i,j>$; An event: a set of transitions

$n=(n_1,\ldots,n_M)$: $n_{+i}=(n_1,\ldots,n_i+1,\ldots,n_M)$; $n_{-i}=(n_1,\ldots,n_i-1,\ldots,n_M)$

An arrival customer accepted: $a_+ = \{< n, n_{+i} >, \text{all } n \ \& \ i\}$

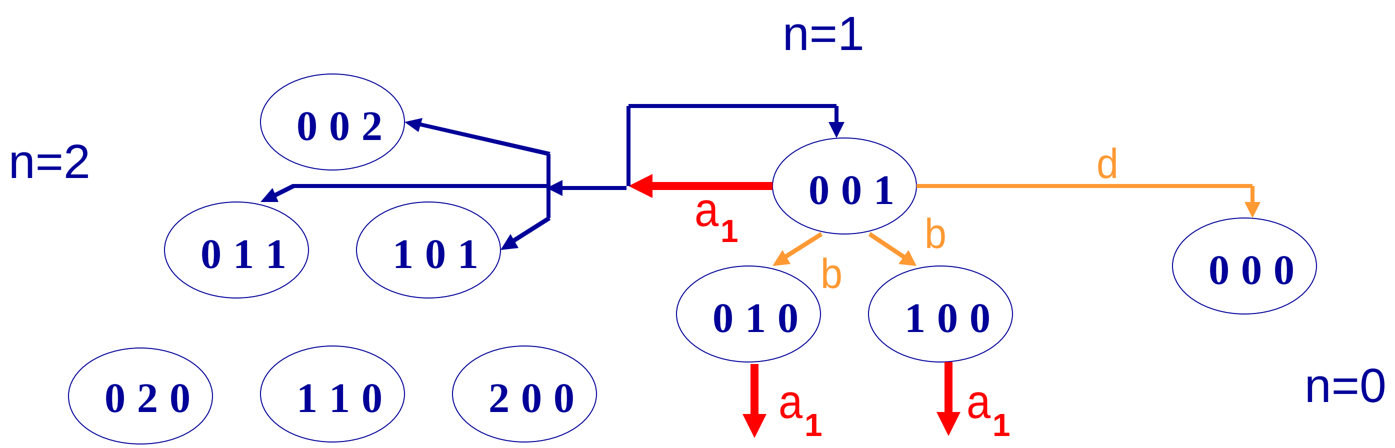
An arrival customer rejected: $a_- = \{< n, n >, \text{all } n\}$

A customer arrival: $a = a_+ U a_-$

A customer departure: $d_- = \{< n, n_{-i} >, \text{all } n \ \& \ i\}$

Three Types of Events

Observable Event: Arriving customer finds population n:



3 svr, 2 cust.

1. We may observe the event
2. Contains some information

Controllable Event:

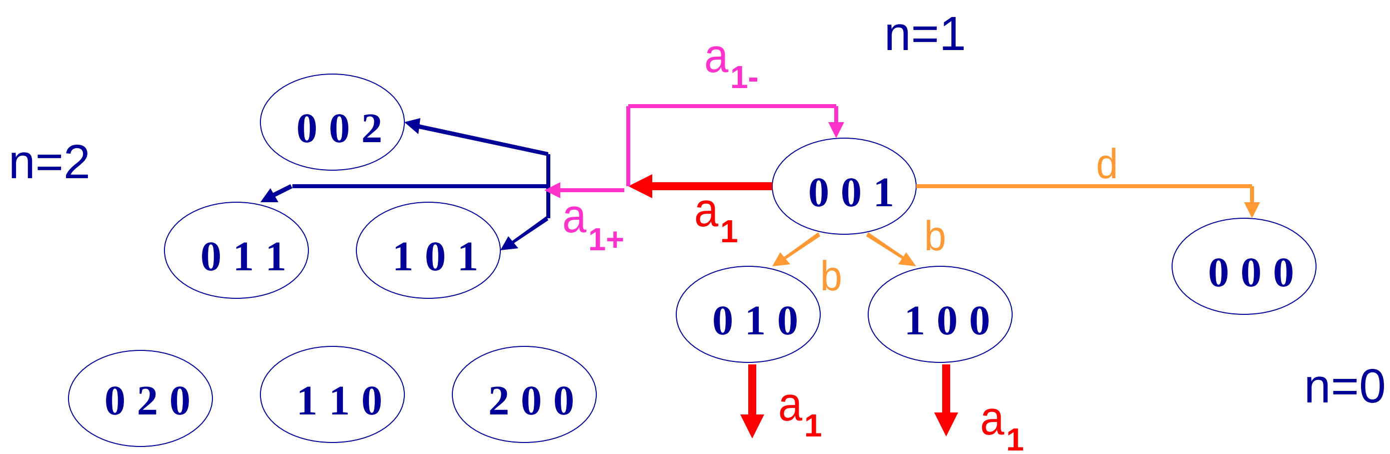
Arriving customer accepted:

$$a_{n,+} = \{< n, n_{+i} >, \text{all } i \,\&\, n \text{ in } S_n\}$$

$$S_n = \{n, \text{with } \Sigma\, n_i = n\ \}$$

Arriving customer rejected:

$$a_{n,-} = \{< n, n>,\ \ n \text{ in } S_n\}$$



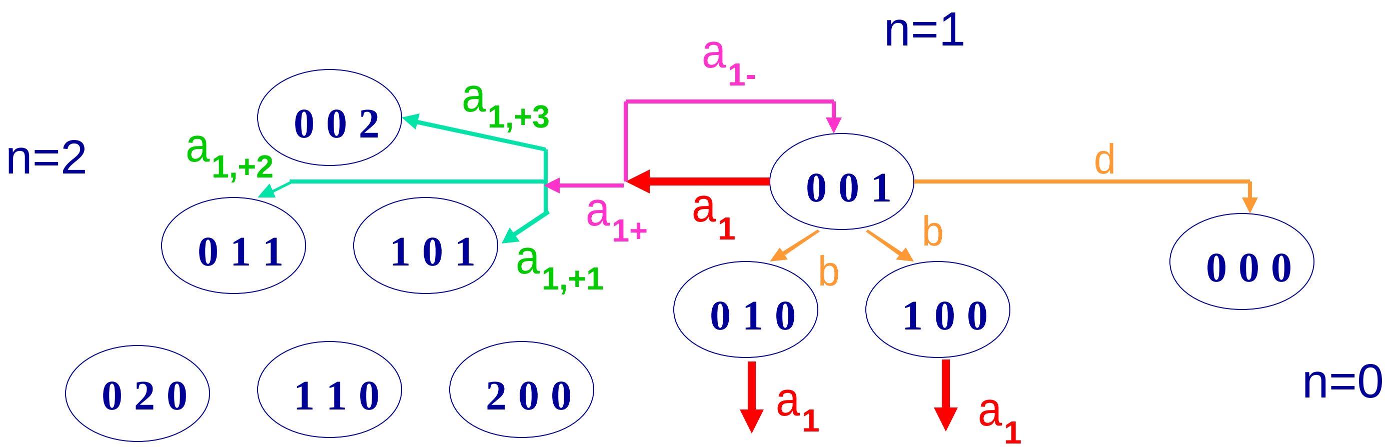
$$a = a_{1+} \cup a_{1-}$$

We can control the probabilities of these events: $b(n) = p(a_{n,+}|a_n)$

Natural Transition Event:

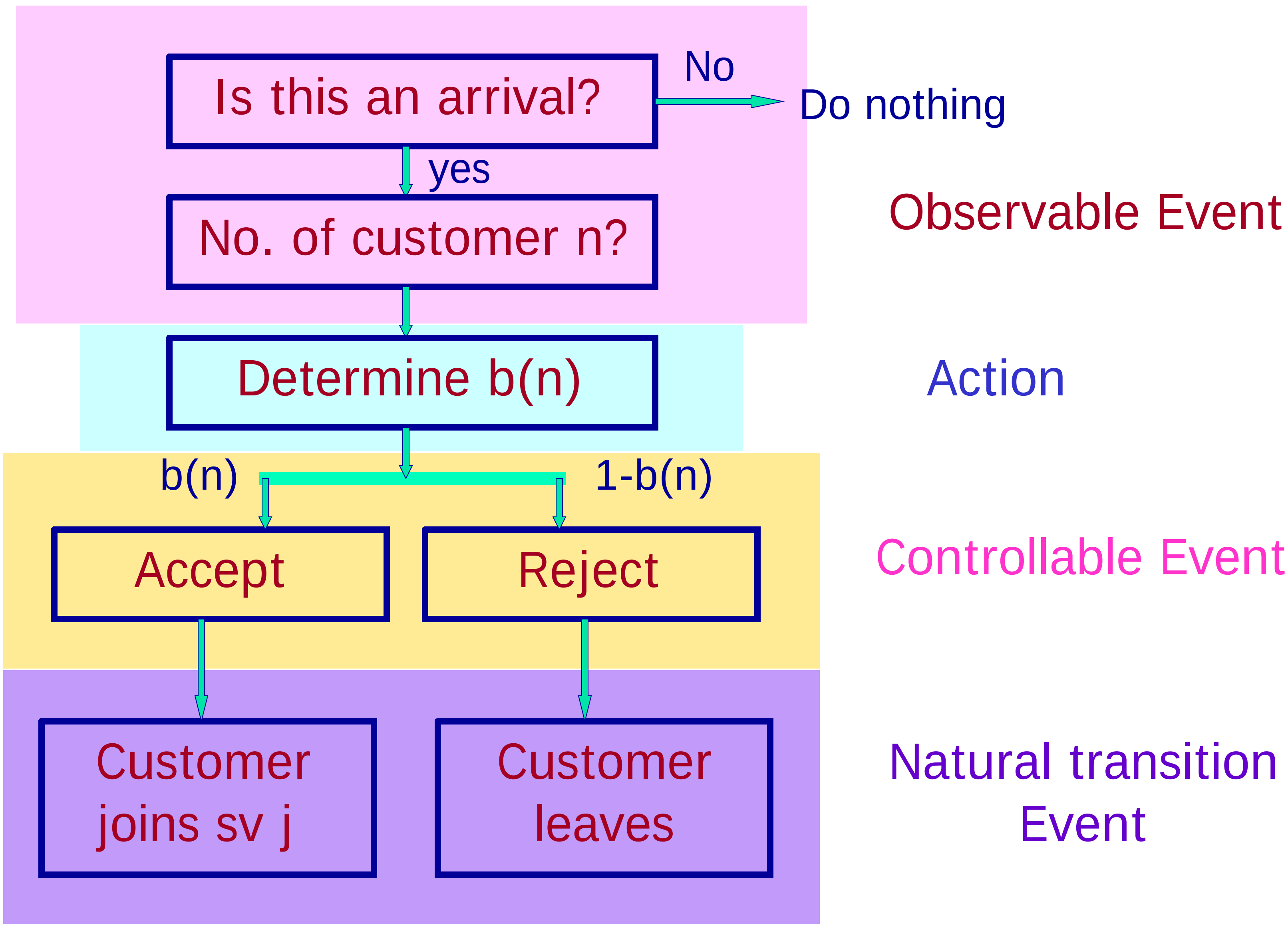
A customer entering server i:

$a_{n,+i} = \{< n, n_{+i} >, n \text{ in } S_n\}$



Nature determines randomly

Logical Relations in Events



Three events, observable, controllable, and natural transition, happens simultaneously in time, but have a logical order

➔ State dependent Markov model does not fit well

Event-Based Optimization

Underlying Markov process $\{X_n , n=0,1,\ldots\}$

At time n, we observe an observable event a_n or b, d (not X_n)

- Determine an action $\mathcal{L}(a_n)$ - controls the prob. of
- Controllable event occurs - a_{n+} or a_{n-}
- Natural transition event follows

Determines State transition

Goal: find a policy L to optimize the performance.

Partially Observable MDP (POMDP)

Underlying Markov process $\{X_n , n=0,1,\ldots\}$

At time n, we observe a random variable Y_n (not X_n)

- Determine an action $\mathcal{L}(Y_0..Y_n)$ - controls state transition prob.
- State transition follows

Goal: find a policy L to optimize the performance.

HOW

To Find a Solution?

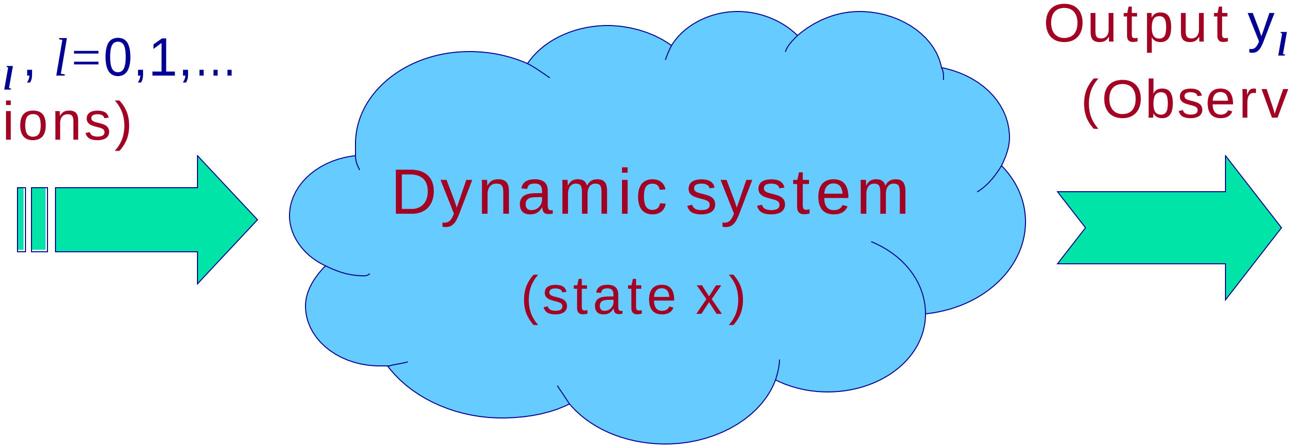
A Sensitivity-Based View of Learning and Optimization

The System

System Performance
η

Inputs α_l, l=0,1,...
(actions)

Output y_l, l=0,1,...
(Observations)



Policy: action= f(information) , $\alpha = L(y)$

L may depend on parameters θ

Learning and Optimization

Optimizatiom: Determine a policy *L* that maximizes η^L

Difficulties:

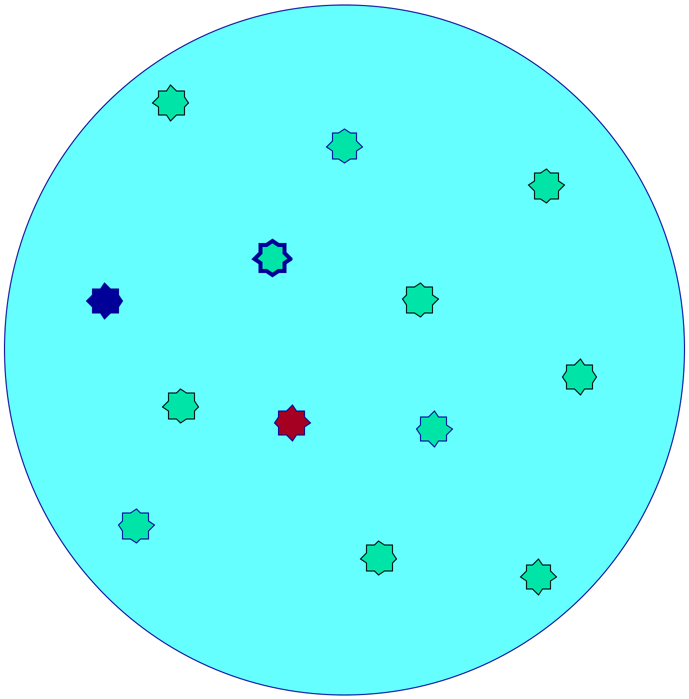
1. Policy space too large (# states: N, # actions: M, # policies: M^N)
2. System too complicated, structure/parameters unknown

Learning:

1. Observing /analyzing sample paths
   (with or without estimating system parameters).
2. Study the behavior of a system under one policy

   to learn how to improve system performance

A Philosophical Boundary



- We can study/observe only one policy at a time
- By studying or observing the behavior of one policy, in general we cannot obtain the performance of other policies

We can do very little!

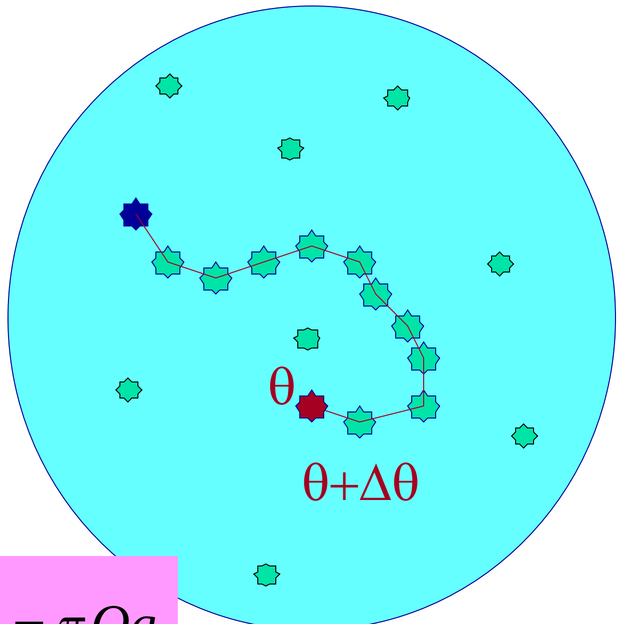
Two Basic Approaches

- Continuous Parameters

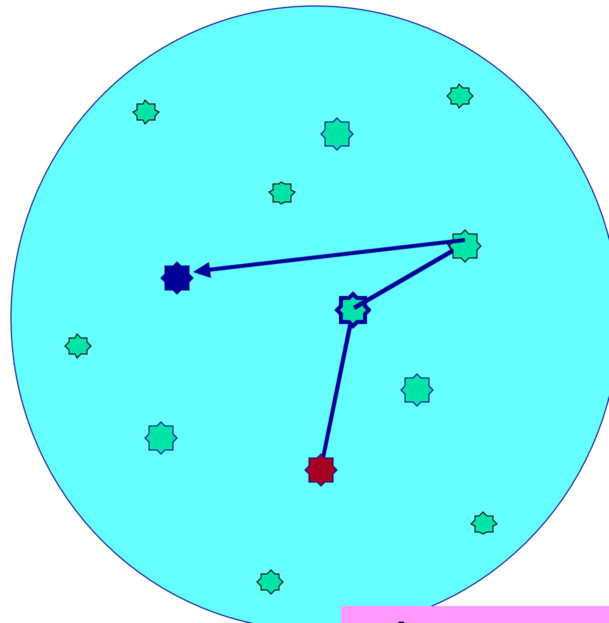
  (policy gradient)

- Discrete Policy Space

  (policy iteration)



$$\frac{d\eta}{d\theta} = \pi Q g$$



$$\eta' - \eta = \pi' Q g$$

We can only start with these two sensitivity formulas!

Performance Difference Formula

For two Markov chains *P*, *f*, η, π and *P'*, *f*, η', π', let *Q=P'-P*

Poisson equation and performance potentials g :

$$(I - P + e\pi)g = f$$

$$\times\pi : \qquad \pi g = \pi f = \eta \qquad \eta' = \pi' f$$

$$\times\pi' : \qquad \eta' - \eta = \pi'(P' - P)g = \pi' Q g$$

Potentials g can be estimated from a sample path of (P,f)

$$g(i) = E\{\sum_{k=0}^{\infty}[f(X_k) - \eta] \mid X_0 = i\}$$

Policy Iteration

$$\eta' - \eta = \pi' Q g = \pi' (P' - P) g$$

1. $\eta' > \eta$ if $P'g > Pg$

2. Policy iteration:

   At any state find a policy P' with $P'g > Pg$

3. Multi-chain (average performance):
   A simple, intuitive approach without discounting

Limitations of State-Based Approaches

- Curse of dimensionality: state space too large
- Independent actions: action at different states need to be independent
- Structural property lost

Event-Based Optimization

Event-based policy: $a_n \rightarrow b(n)$

Problem: Find a policy that maximizes steady-state perf.

Method: Start with perf. difference formula

How to derive PDF?

Construction method with performance potentials as building blocks.

Performance Difference Formula

Two policies: b(n) and b'(n)

1. PDF: $$\eta' - \eta = \sum_{n=0}^{N-1} \{\pi'(n)[b'(n) - b(n)]d(n)\}$$

π(n): prob. of event a_n , arrival finding population n

Potential aggregation:

$$d(n) = \frac{1}{\pi(n)}\{\sum_{i=1}^{M} q_{0i}[\sum_{\sum n_i = n} p(\bar{n})g(\bar{n}_{+i})] - \sum_{\sum n_i = n} p(\bar{n})g(\bar{n})\}$$

2. Performance deriv: $$\frac{d\eta}{d\theta} = \sum_{n=0}^{N-1} \{\pi(n)\frac{db_\theta(n)}{d\theta}d(n)\}$$

1➔ policy iteration: Reduced computation: N linear in size
action depends on states

2➔ gradient-based optimization (Perturbation analysis)

3. Learning and On-line implementation:
d(n) estimated with same amount of computation as g(n)!

Summary:

Why ?

- Action taken when an event happens
- Same event may happen at different states,
  i.e., actions may be the same for different states

How ?

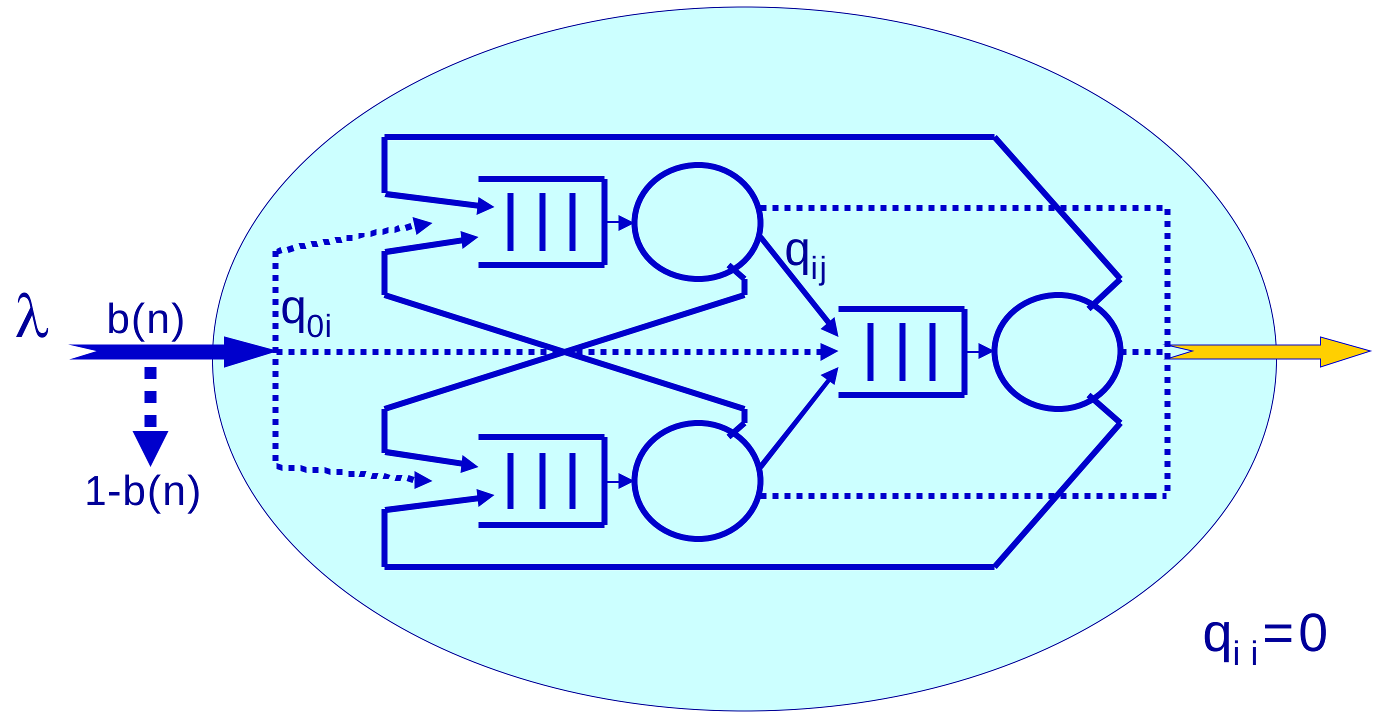
- Event-based policy
- Perf. diff. formula for event based policies
- Utilize system structure (aggregation)

Benefit ?

- Event-based policy iteration possible
- Reduce computation (e.g, linear in system size)
- Intuitive, clear physical meaning

Other Examples

Example 1: Admission Control in Communication



n: number of all customers in network

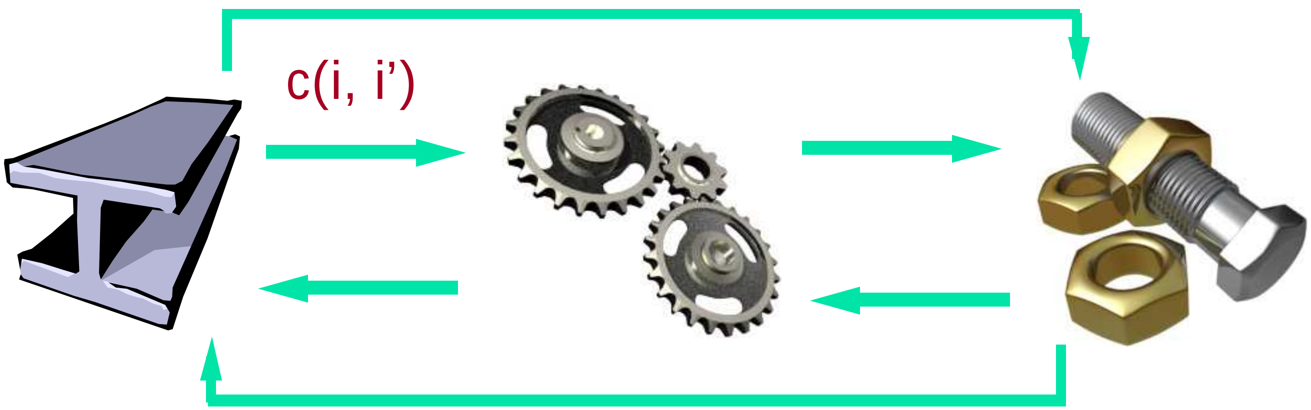
n_i: number of customers at server i

$n=(n_1,\ldots,n_M)$: state, N: Capacity

How do we choose the admission probability b(n)?

Events: A customer arrives finding a population n

Example 2: Manufacturing



M products: $i=1,2,\ldots, M$

Operation of product i consists of N_i phases, $j=1,\ldots,N_i$

State: (i,j)

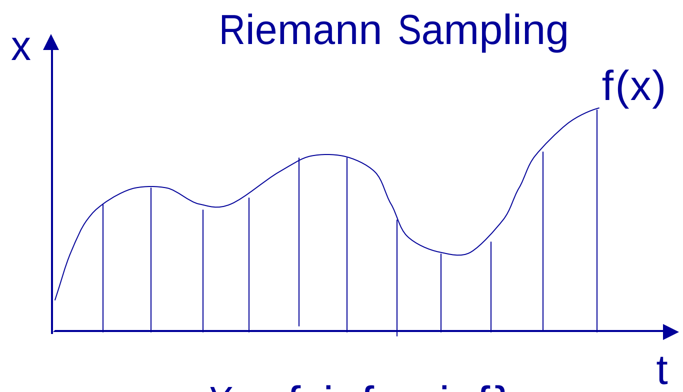
Two level hierarchical control

When a product i is finishes, how do we choose

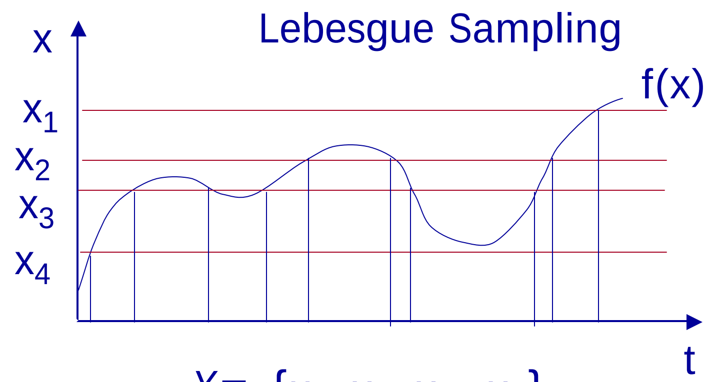
The next product i', (probability c(i,i'))?

Events: product i finished

Example 3: Control with Lebesgue Sampling



X= {-inf, +inf}



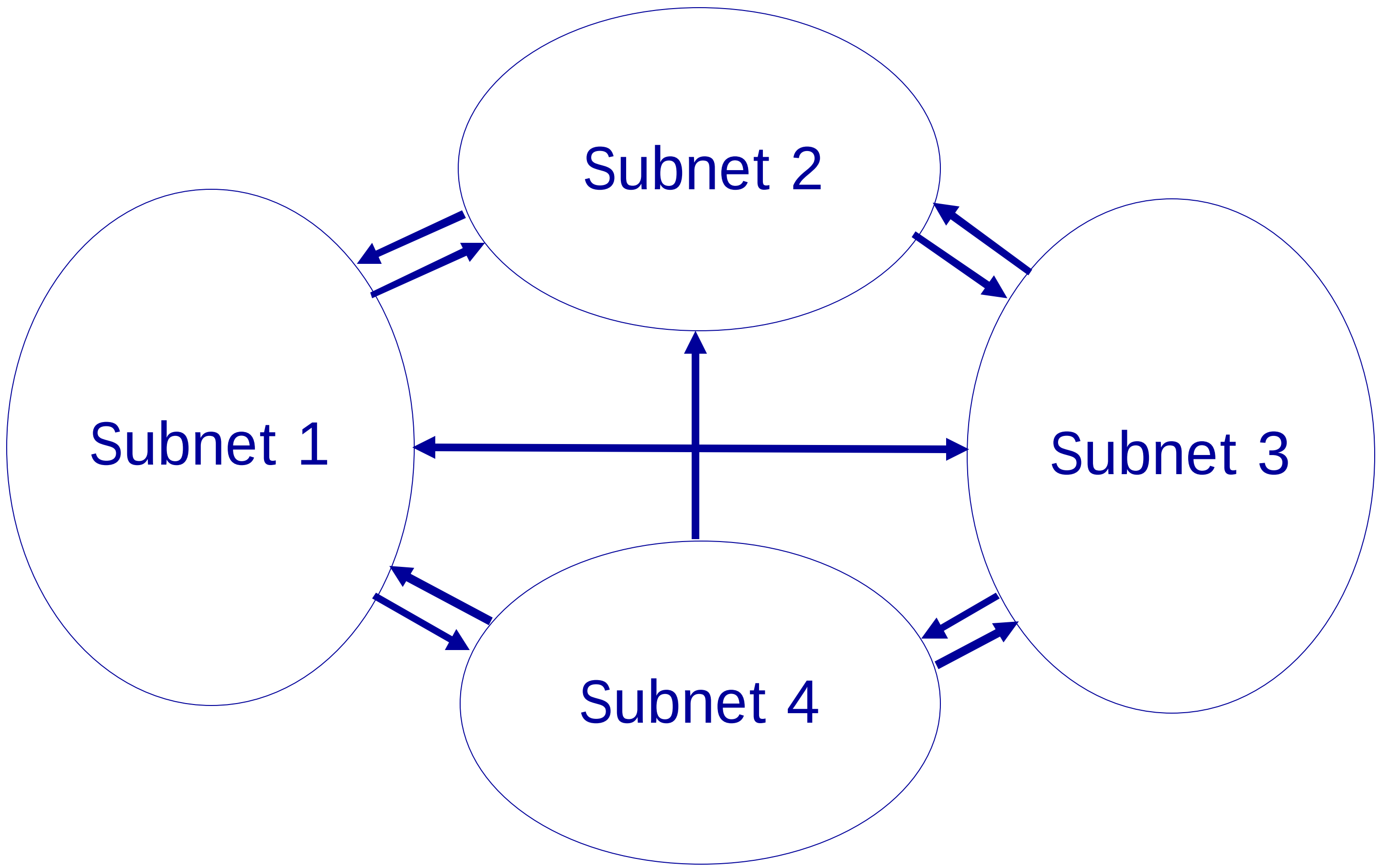
$X = \{x_1, x_2, x_3, x_4\}$

LSC More efficient than RSC (Astrom, et al, for $X=\{x_1\}$)

How to develop an optimal control policy for LSC with $X = \{x_1, \ldots\}$?

Events: System reaches a state in $X = \{x_1, \ldots\}$

Example 4: Large Communication Networks



Conclusion and Discussion

- Learning and optimization are bases on two performance sensitivity formulas
- They lead to two fundamental approaches:
  gradient based (PA) and policy iteration based
- State-based model has limitations:
  State space too large, action independent, no structure
- Event-based optimization
  - Event: a set of transitions, extension of set of states
  - Three types of events, logically related, structure properties
  - Solution based on PDF
  - Computation reduced, maybe linear in system size
  - On-lime implementation
  - Widely applicable
    (Exs: some not fit Markov model, others utilize special features)

Thank You!